Aix*Marseille université

LIS
LABORATOIRE D'INFORMATIQUE & SYSTÈMES

Deep generative models

→ VAE, GANs, Diffusion

Pourquoi ?

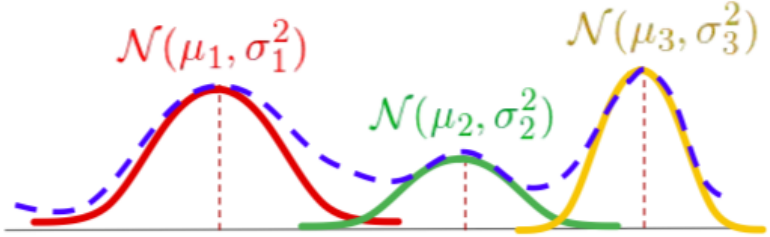
- Estimation de densité de probabilité P(X) à partir d'observations X
- P(Y|X) vs P(X) ou P(X,Y), ie: supervisé vs non-supervisé

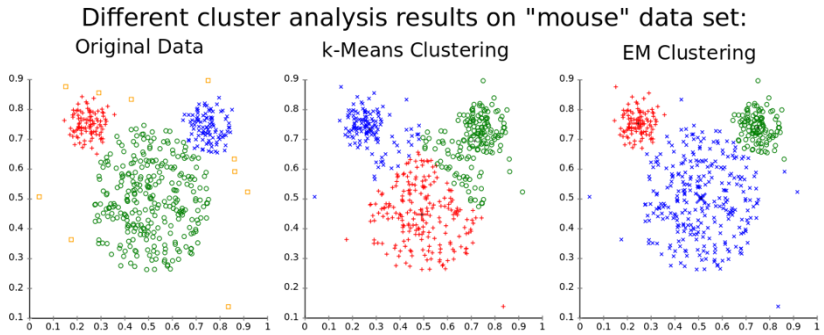
- Générer de nouvelles données
  - Utile lorsque peu de données disponibles
  - Compléter des données manquantes

- Nombreuses applications
  - Adaptation, transfert, multivues, …
  - Interpolation (dans un espace latent)

Pourquoi ?

- Partitionnement de données (ie: GMM)
  - Estimation d'une distribution de probabilité pour chaque cluster
  - Estimer P(X) comme une somme pondérée de gaussiennes : $p(x) = \sum_{i=1}^{k} w_i p_i(x)$
- Réduction de dimension (ie: PPCA, FA)
  - Trouver P(Z|X) ~ $\mathcal{N}(\mu, \Sigma)$
- Apprentissage non supervisé
  - Pas besoin d'étiquette, seulement les données !





Apprentissage non supervisé

- Données : X , Z
  - Données observées x_1, … x_N
  - Variables latentes z_1, …, z_k

- Objectif : Déterminer une structure sous jacente de X
  - Apprendre une distribution sur les variables latentes Z
  - $$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Apprentissage non supervisé

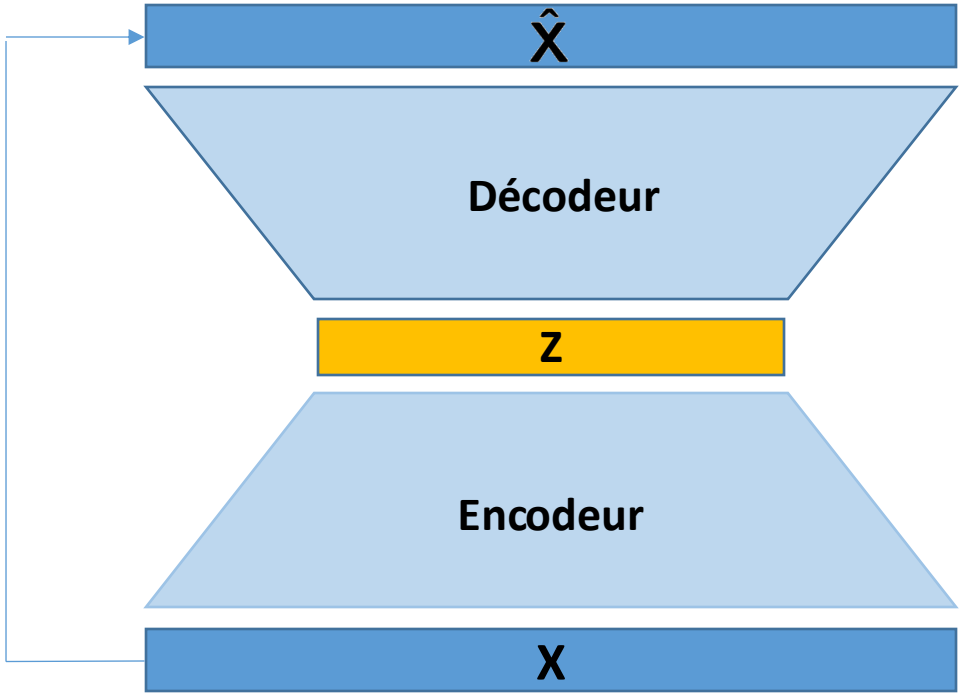
- Données : X , Z
  - Données observées x_1, … x_N
  - Variables latentes z_1, …, z_k

- Objectif : Déterminer une structure sous jacente de X
  - Apprendre une distribution sur les variables latentes Z
  - $$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Compliqué… alors le deep learning peut aider !

Retour sur les Autoencodeurs

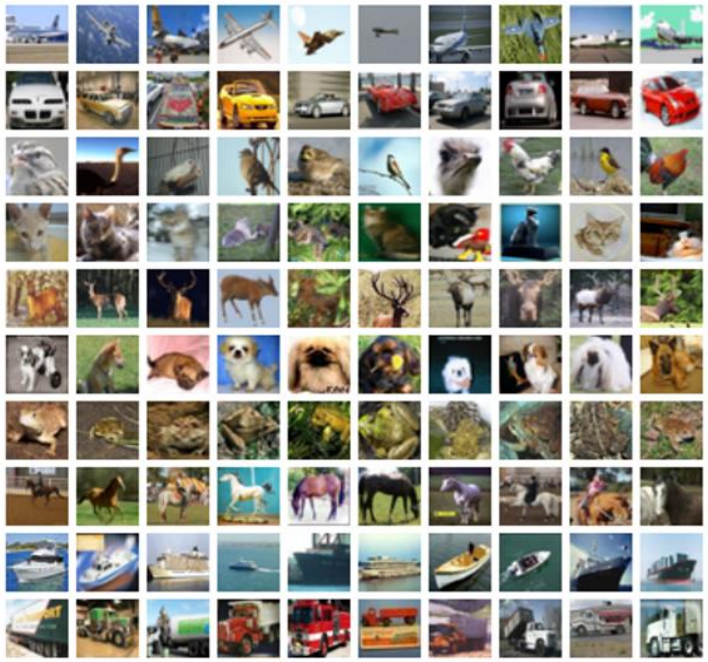
- Architecture dédiée pour l'apprentissage d'un espace latent de manière non supervisée
- Objectif : $min \left\| X - \hat{X} \right\|^2$
- Deux entités apprises conjointement
  - Encodeur qui modélise $P(Z|X)$
  - Décodeur qui modélise $P(\hat{X}|Z)$

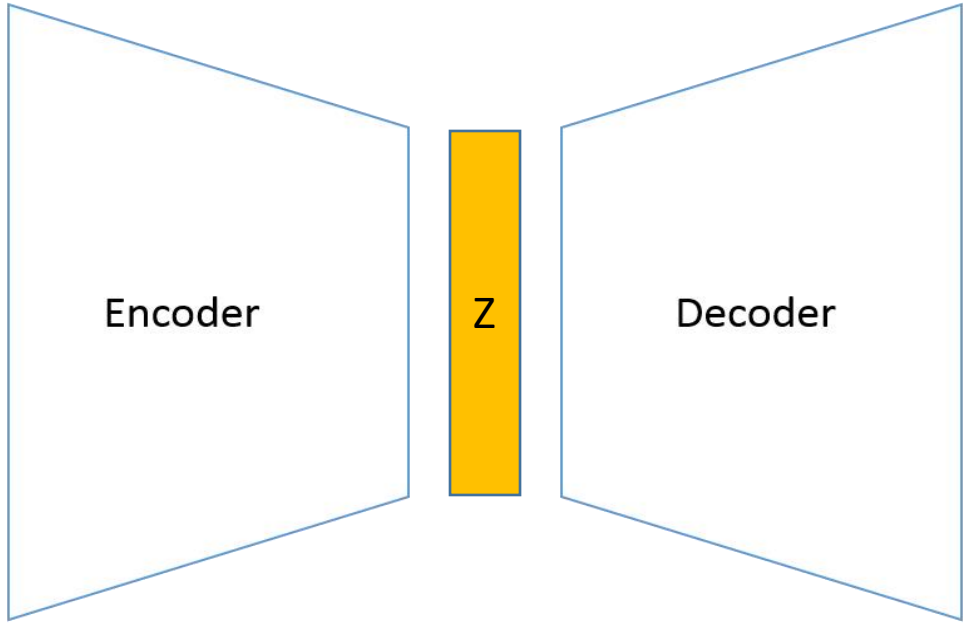


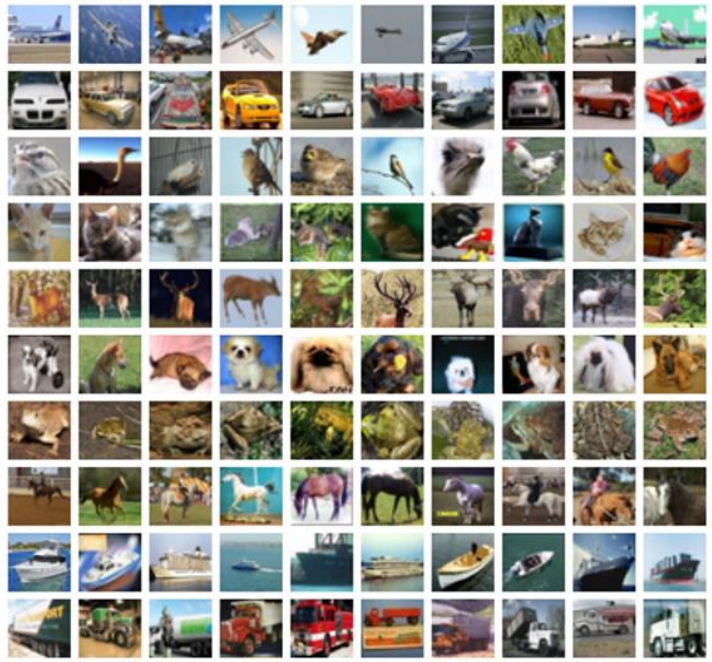
- Intérêt en compression, apprentissage de features, débruitage, …

Retour sur les Autoencodeurs

- Exemple avec des images naturelles 32 x 32 x 3 (CIFAR10)







- Espace latent de dimension 4 x 4 x 16 = 256

Retour sur les Autoencodeurs

- Problème : peu utile pour la génération (d'images) …

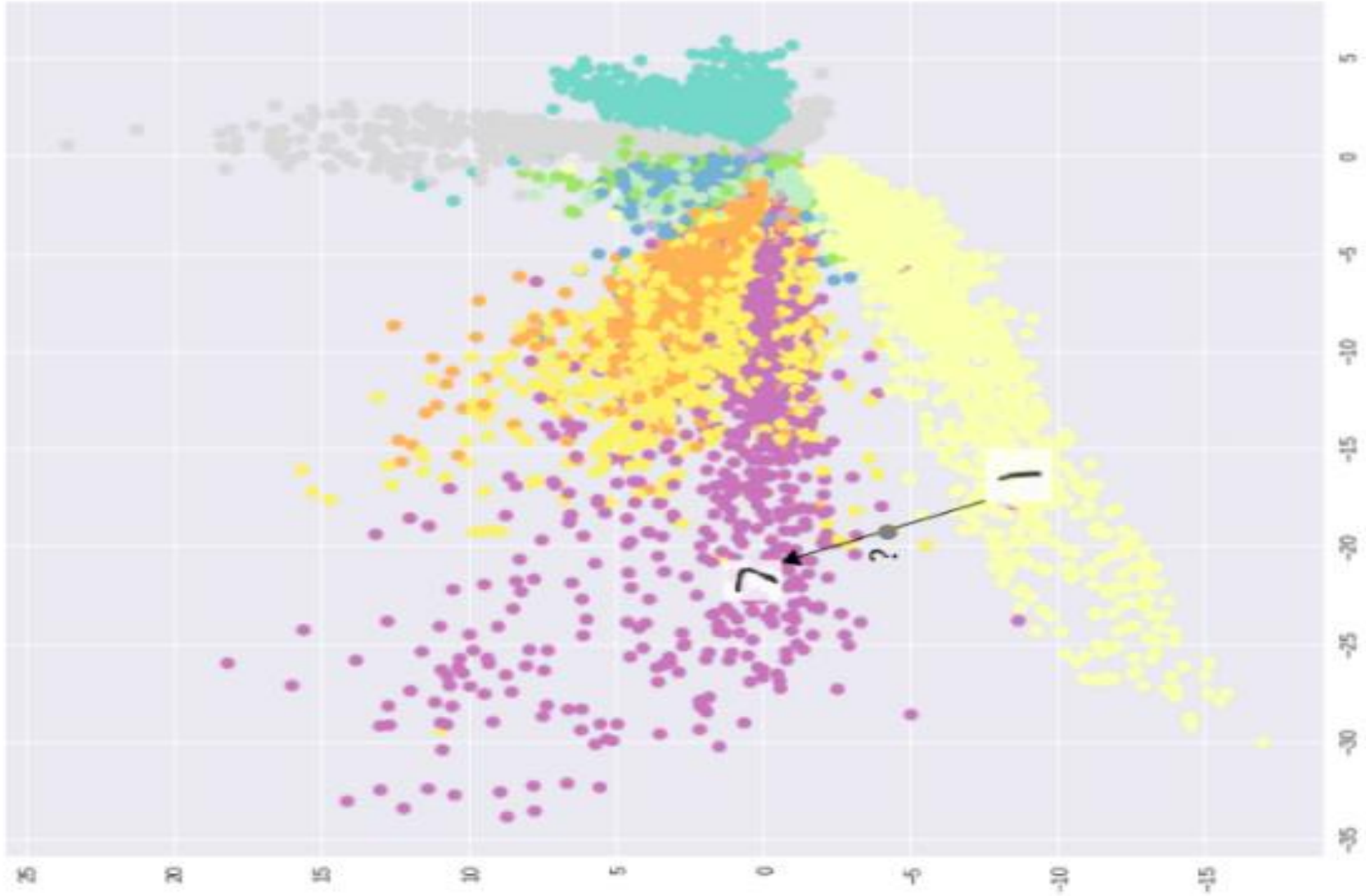
- La fonction de perte L2 n'est pas adaptée aux distributions multimodales

→ Tend à générer du flou

- Pas de contrainte sur la structure de l'espace latent

→ Peu compact ou non continu : interpolation difficile

Retour sur les Autoencodeurs



Variational Autoencoder

- [Kingma, 2014]
- Cherche à maximiser $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$
- Calcul insoluble si Z est continue et/ou de très grande dimension
  - Idée : introduire *P(Z|X)* et l'approximer par une distribution Q(Z|X) plus simple
  - Minimiser la divergence entre *Q(Z|X)* et *P(Z|X)*
- Evidence Lower Bound (ELBO)
  - $\mathtt{L}(X, P_\theta, Q_\phi) < \mathtt{L}(X, P_\theta)$

Variational Autoencoder

Minimiser $KL = KL(Q(Z \mid X) \mid\mid P(Z \mid X))$

$$
\begin{aligned}
&= \int_z \left(Q(Z \mid X) \log \frac{Q(Z \mid X)}{P(Z \mid X)} \right) dz \\
&= \int_z \left(Q(Z|X) \left(\log Q(Z|X) - \log P(Z|X) \right) \right) \\
&= \int_z \left(Q(Z|X) \left(\log Q(Z|X) - \log \frac{P(Z,X)}{P(X)} \right) \right) \\
&= \int_z \left(Q(Z|X) \left(\log Q(Z|X) - \log P(Z,X) + \log P(X) \right) \right) \\
&= \int_z \left(Q(Z|X) \left(\log Q(Z|X) - \log P(X|Z) - \log P(Z) + \log P(X) \right) \right) \\
&= -\int_z Q(Z|X) \log P(X|Z) + \int_z Q(Z|X) \log \frac{Q(Z|X)}{P(Z)} + \log P(X) \\
&= -\mathbb{E}_{z \sim Q(Z|X)} \log P(X|Z) + KL(Q(Z|X) \mid\mid P(Z)) + \log P(X)
\end{aligned}
$$

Variational Autoencoder

Reconstruction

$$\log P(X) = \mathbb{E}_z \log P(X|Z) - KL(Q(Z|X) \,||\, P(Z)) + KL(Q(Z|X) \,||\, P(Z|X))$$

ELBO

≥ 0

- Maximiser $\mathbb{L}(X, P_\theta)$ ⇔ Maximiser la reconstruction et minimiser la KL
- $$\mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})] = -\frac{1}{2}\sum_{k=1}^{K}\{1 + \log\sigma_k^2 - \mu_k^2 - \sigma_k^2\}$$ si $P(Z) \sim \mathcal{N}(0, \mathrm{I})$
- $Q(Z|X)$ est un encodeur qui apprend $\mu_{z|X}$ et $\sigma_{z|X}$
- $P(X|Z)$ est le décodeur sachant le prior Z (+ reparameterization trick)

Variational Autoencoder

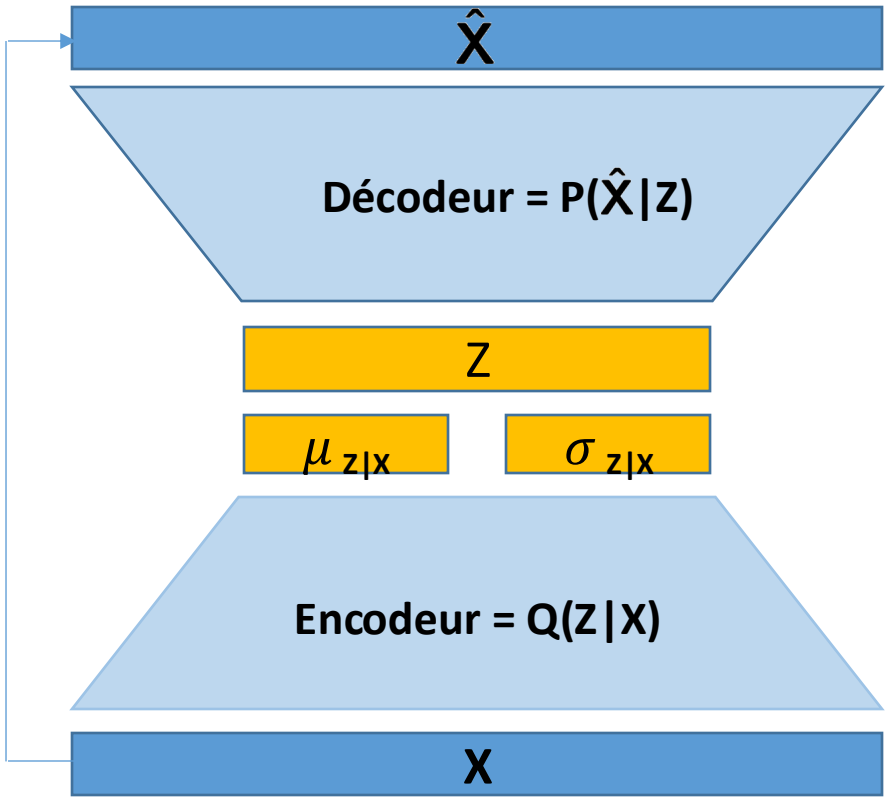
- Sampler Z directement sur $\mathcal{N}(\mu_{z|x}, \sigma_{z|x})$ empêche l'apprentissage par backprop

$\Rightarrow$ Reparameterization trick

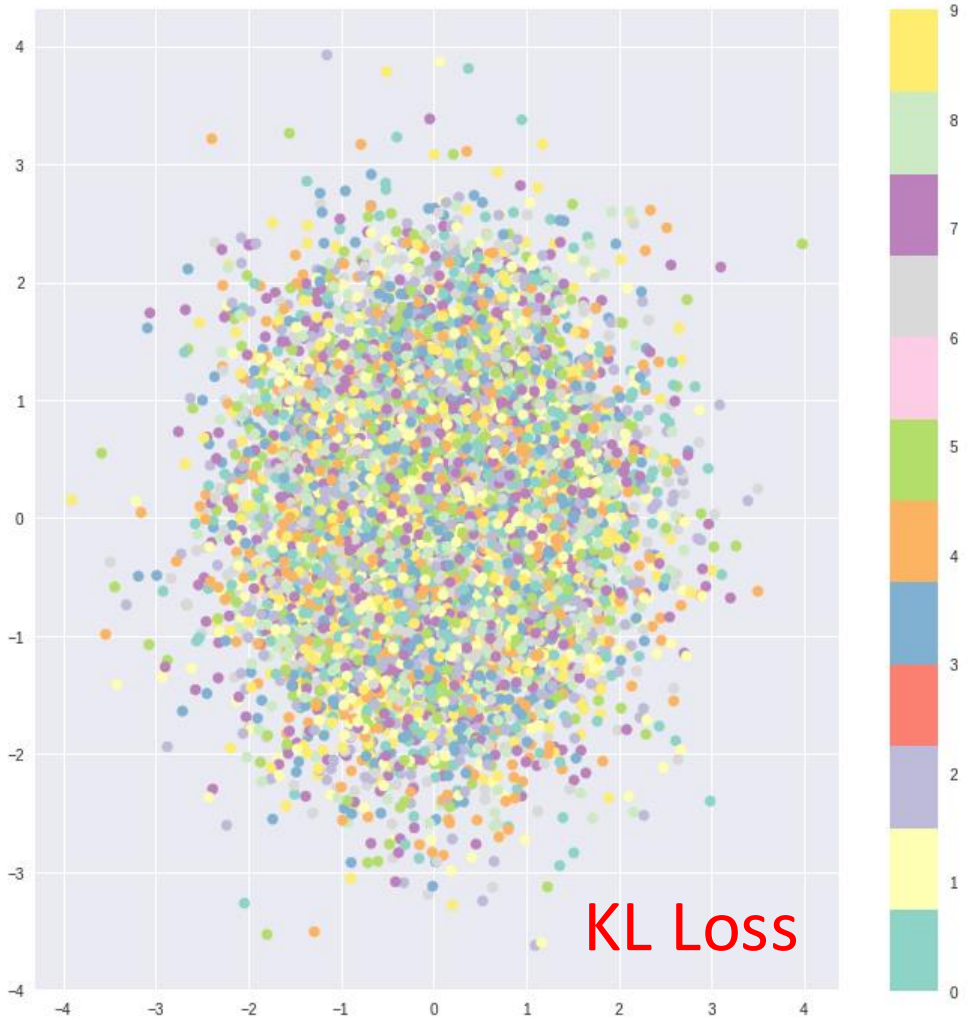
$z = \mu_{z|x} + \varepsilon\, \sigma_{z|x}^2$ avec $\varepsilon \sim \mathcal{N}(0, I)$

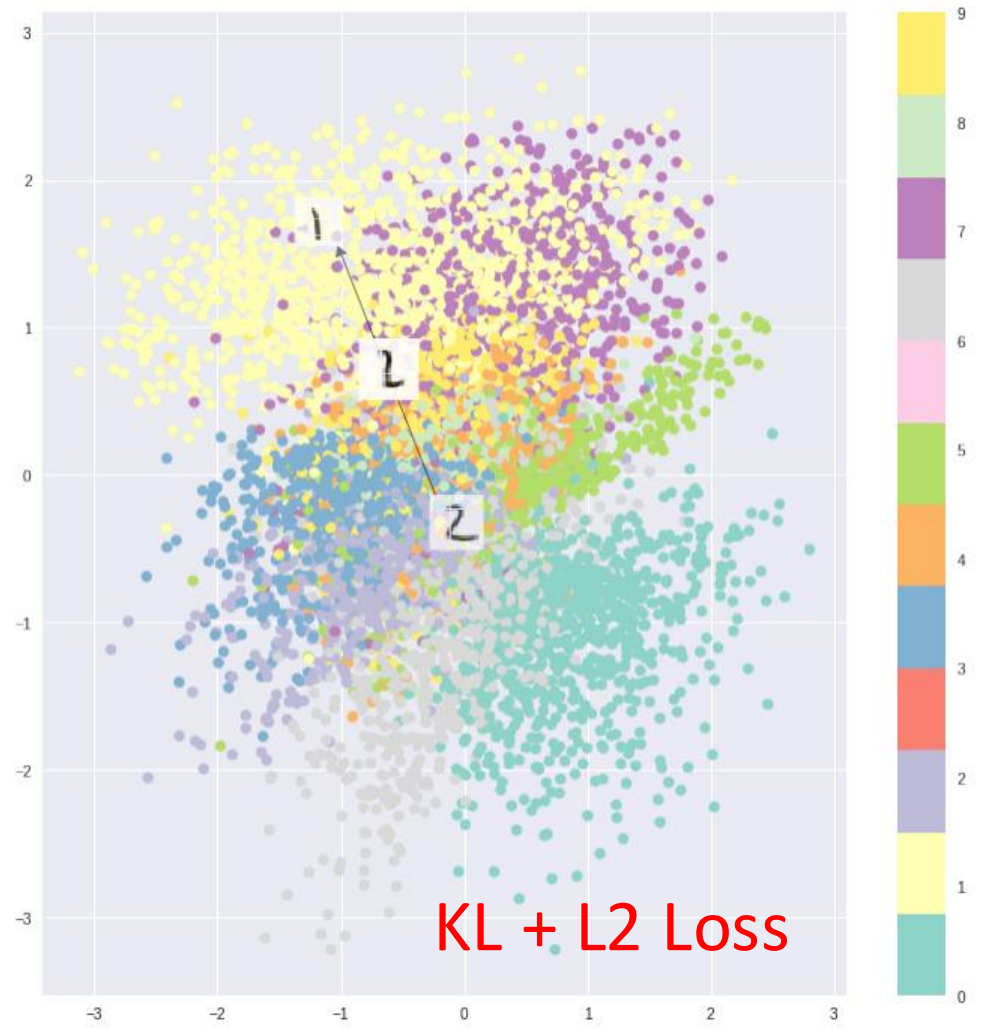
- Après convergence, $\mu \approx 0$ et $\sigma \approx I$

$\Rightarrow$ Génération possible à partir de $\mathcal{N}(0, I)$!



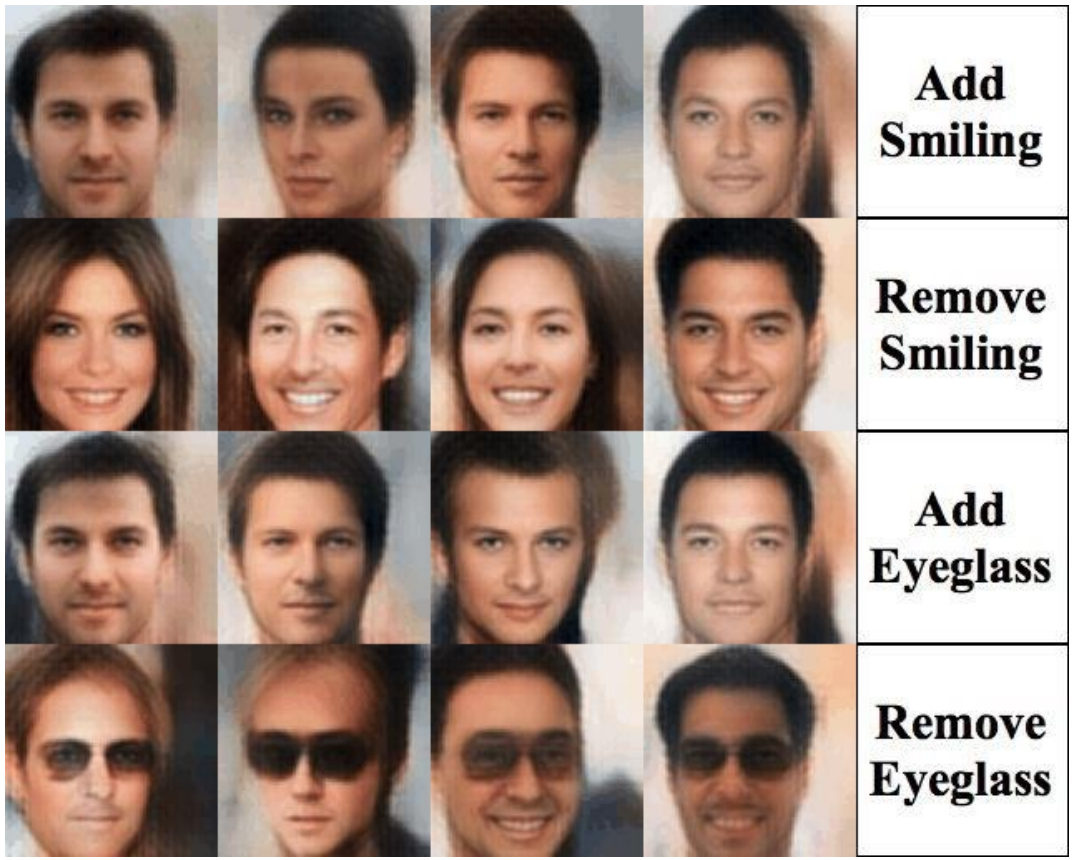
Variational Autoencoder



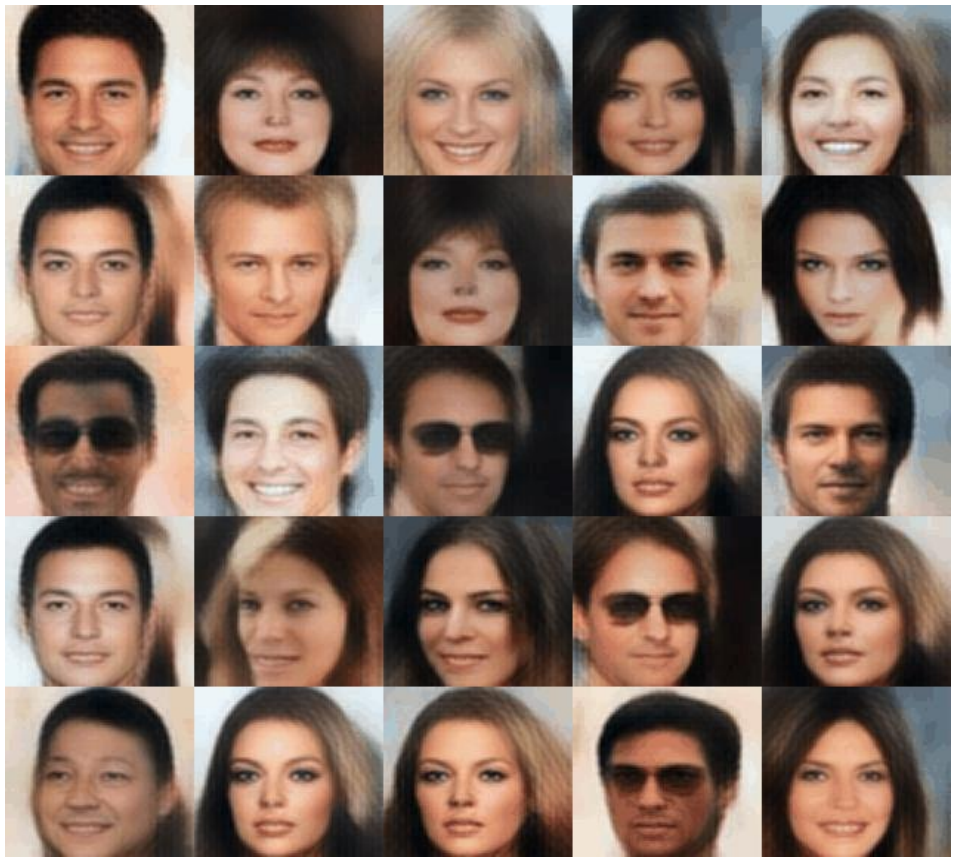


Variational Autoencoder

Arithmétique dans l’espace latent



Interpolation entre exemples

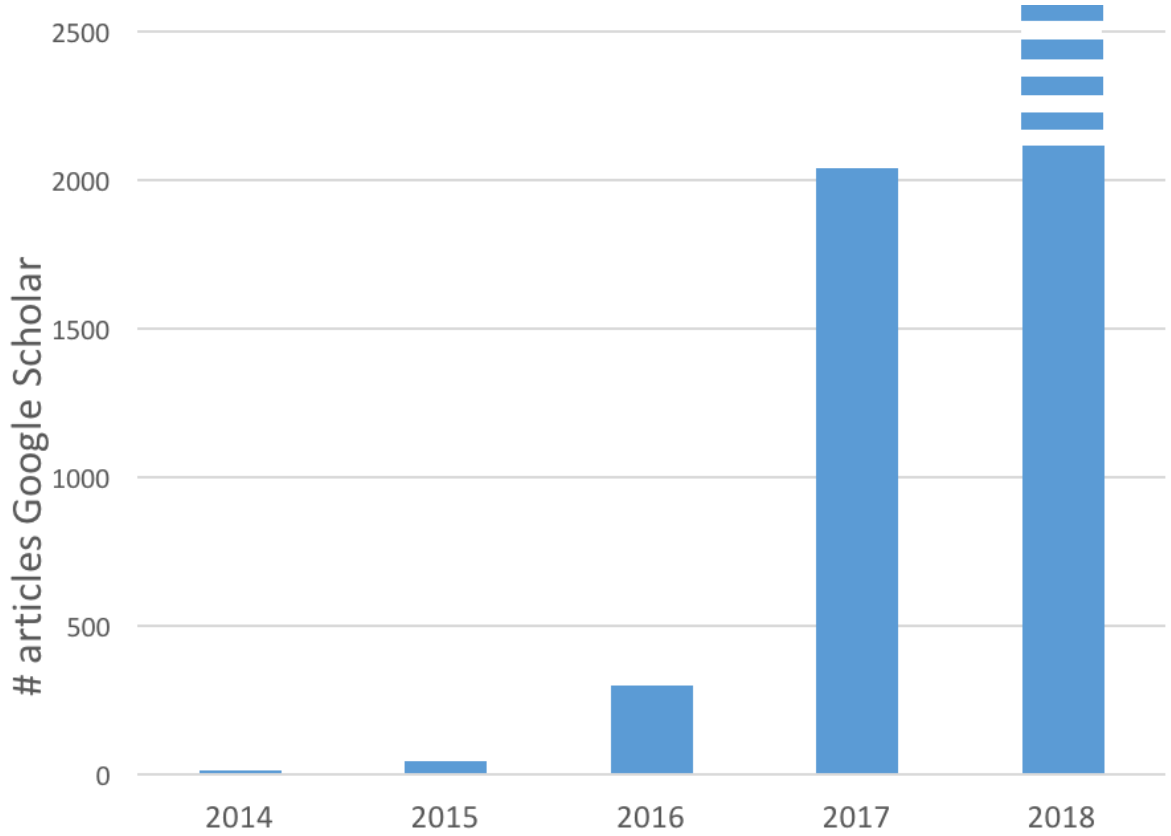


Generative Adversarial Network

- Paradigme d'apprentissage récent [Y. Goodfellow, 2014]
  - *Adversarial training is the coolest thing since sliced bread. — Yan LeCun*

- Utile pour apporter une supervision lorsque pas disponible

- Enormément de travaux dérivés depuis le premier papier



Generative Adversarial Network

- Un « jeu » entre deux adversaires, objectifs contradictoires

Un faussaire veut produire des billets de banque les plus réalistes. Risque la prison

Un policier essaie d'attraper des faux billets. Pénalité s'il se trompe

Generative Adversarial Network

- Un « jeu » entre deux adversaires, objectifs contradictoires

générateur = de billets

Un faussaire veut produire des billets de banque les plus réalistes. Risque la prison

Un policier essaie d'attraper des faux billets. Pénalité s'il se trompe

= classifieur : billet → vrai/Faux

Generative Adversarial Network

- Un « jeu » entre deux adversaires, objectifs contradictoires

Un faussaire veut produire des billets de banque les plus réalistes. Risque la prison

Un policier essaie d'attraper des faux billets. Pénalité s'il se trompe

Generative Adversarial Network

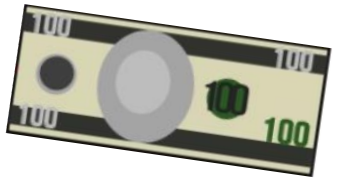
- Un « jeu » entre deux adversaires, objectifs contradictoires

Un faussaire veut produire des billets de banque les plus réalistes. Risque la prison

Un policier essaie d'attraper des faux billets. Pénalité s'il se trompe

Generative Adversarial Network

- Un « jeu » entre deux adversaires, objectifs contradictoires



Un faussaire veut produire des billets de banque les plus réalistes. Risque la prison

Un policier essaie d'attraper des faux billets. Pénalité s'il se trompe

Generative Adversarial Network

- Un « jeu » entre deux adversaires, objectifs contradictoires

Un faussaire veut produire des billets de banque les plus réalistes. Risque la prison

Un policier essaie d'attraper des faux billets. Pénalité s'il se trompe

Generative Adversarial Network

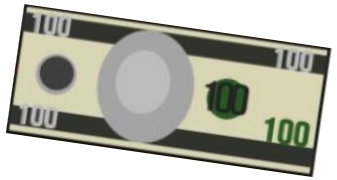
- Un « jeu » entre deux adversaires, objectifs contradictoires

Un faussaire veut produire des billets de banque les plus réalistes. Risque la prison

Un policier essaie d'attraper des faux billets. Pénalité s'il se trompe

Generative Adversarial Network

- Un « jeu » entre deux adversaires, objectifs contradictoires



Un faussaire veut produire des billets de banque les plus réalistes. Risque la prison

Un policier essaie d'attraper des faux billets. Pénalité s'il se trompe

Generative Adversarial Network

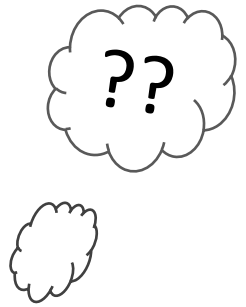
- Un « jeu » entre deux adversaires, objectifs contradictoires

Un faussaire veut produire des billets de banque les plus réalistes. Risque la prison

Un policier essaie d'attraper des faux billets. Pénalité s'il se trompe

Generative Adversarial Network

- Un « jeu » entre deux adversaires, objectifs contradictoires

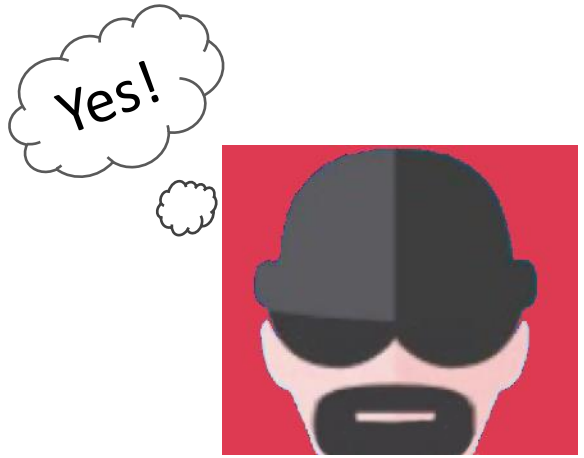


Un faussaire veut produire des billets de banque les plus réalistes. Risque la prison

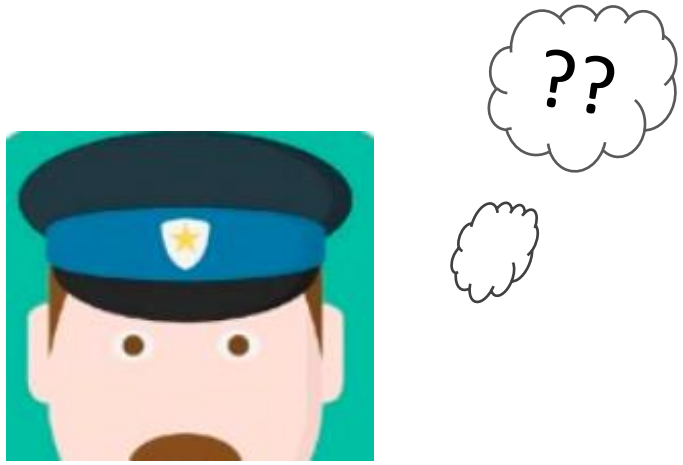
Un policier essaie d'attraper des faux billets. Pénalité s'il se trompe

Generative Adversarial Network

- Un « jeu » entre deux adversaires, objectifs contradictoires



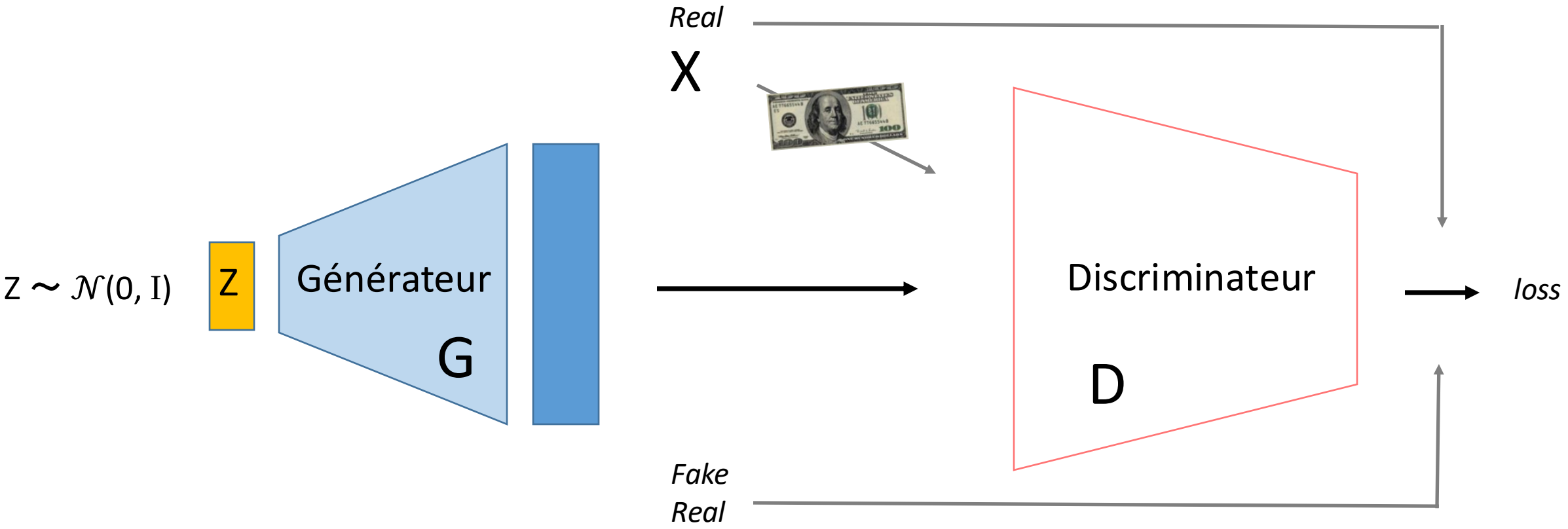
Un faussaire veut produire des billets de banque les plus réalistes. Risque la prison



Un policier essaie d'attraper des faux billets. Pénalité s'il se trompe

Generative Adversarial Network

Données : *X*
Générateur : *P(X|Z)*
Discriminateur : *P({Real|Fake} | X)*



GAN – Objective function

$$\min_G \max_D \mathbb{E}_{x \sim P_X} \log [D(x)] + \mathbb{E}_{z \sim P_Z} \log [1 - D(G(z))]$$

- P_X : Distribution des données (ie: exemples réels) → *P(X)*
- P_Z : Distribution prior de l'espace *Z* → *P(Z)*

GAN – Objective function

$$\min_G \max_D \mathbb{E}_{x \sim P_X} \log [D(x)] + \mathbb{E}_{z \sim P_Z} \log [1 - D(G(z))]$$

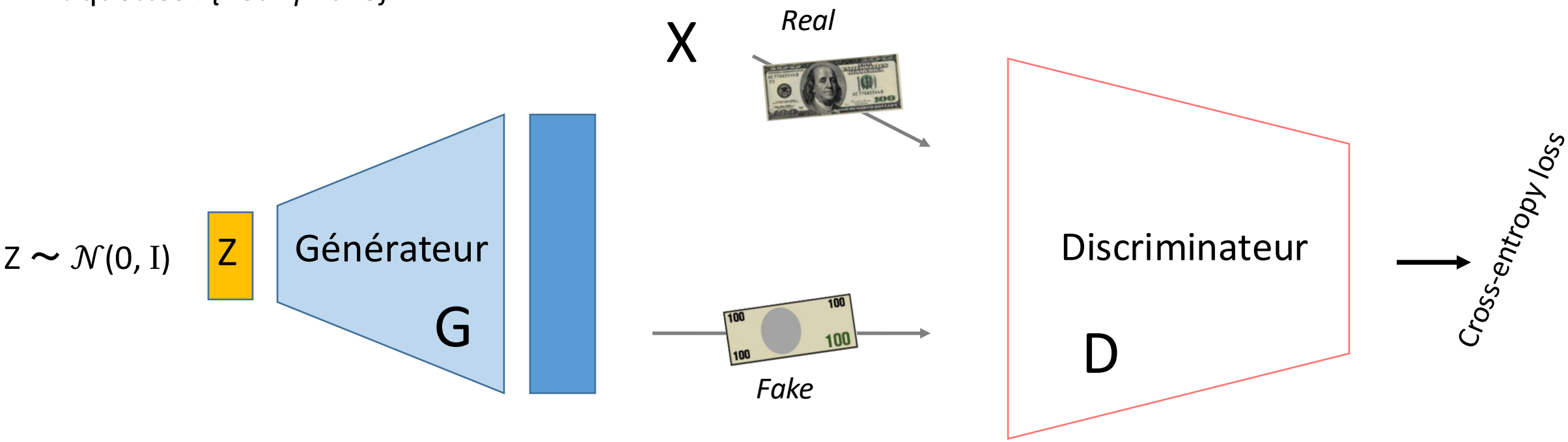
$$\min_G \max_D \mathbb{E}_{x \sim P_X} \log [D(x)] + \mathbb{E}_{x \sim P_G} \log [1 - D(x)]$$

- P_X : Distribution des données (ie: exemples réels) → *P(X)*
- P_Z : Distribution prior de l'espace *Z* → *P(Z)*
- P_G : Distribution apprise par le générateur → *P(X|Z)*

Generative Adversarial Network

- Entrainement du Discriminateur

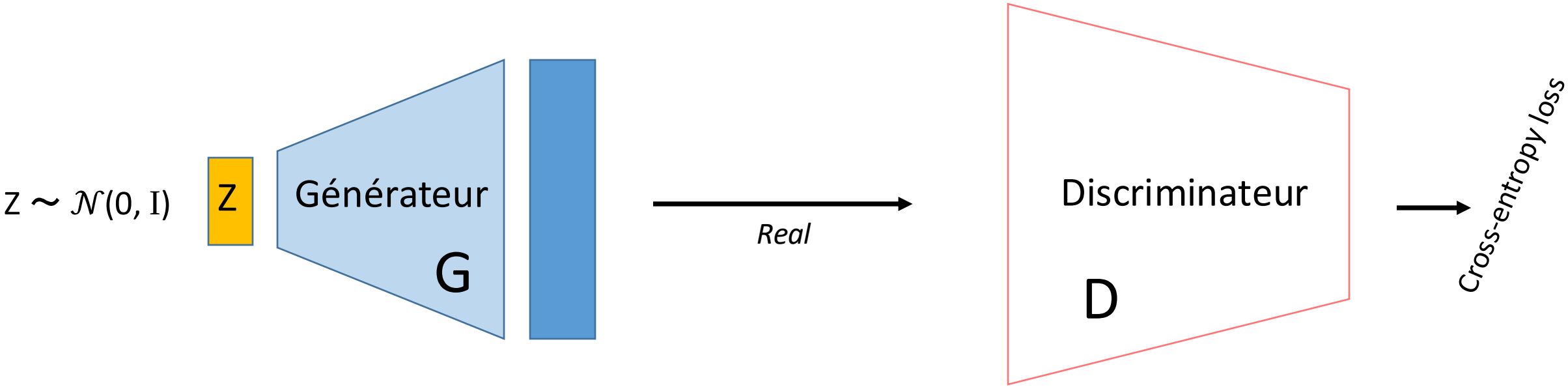
Données : *X*
Etiquettes : *{Real | Fake}*



Generative Adversarial Network

- Entrainement du Générateur

Données : *Z*
Etiquettes : *Real*
D n'est pas mis à jour



Generative Adversarial Network

- Convergence et optimalité : maximisation sur D

$$\mathbb{E}_{x\sim P_X} \log\left[D(x)\right] + \mathbb{E}_{x\sim P_G} \log\left[1 - D(x)\right]$$

$$\int_x P_X(x) \log\left[D(x)\right] dx + \int_x P_G(x) \log\left[1 - D(x)\right] dx$$

$$\int_x P_X(x) \log\left[D(x)\right] + P_G(x) \log\left[1 - D(x)\right] dx$$

maximum de a log(x) + b log(1-x) = a / (a + b)

- Quelque soit G, le meilleur D est $D^*(x) = \frac{P_X(x)}{P_X(x) + P_G(x)}$

Generative Adversarial Network

- Convergence et optimalité : minimisation sur G

$$\mathbb{E}_{x\sim P_X} \log\left[D^*(x)\right] + \mathbb{E}_{x\sim P_G} \log\left[1 - D^*(x)\right]$$

$$\mathbb{E}_{x\sim P_X} \log\left[\frac{P_X(x)}{P_X(x)+P_G(x)}\right] + \mathbb{E}_{x\sim P_G} \log\left[1 - \frac{P_X(x)}{P_X(x)+P_G(x)}\right]$$

$$\mathbb{E}_{x\sim P_X} \log\left[\frac{P_X(x)}{P_X(x)+P_G(x)}\right] + \mathbb{E}_{x\sim P_G} \log\left[\frac{P_G(x)}{P_X(x)+P_G(x)}\right]$$

- Si minimum atteint quand $P_X = P_G$ ➔ $D^*(x) = \frac{P_X(x)}{P_X(x)+P_G(x)} = \frac{1}{2}$

➔ Converge vers $-\log 4$!

Generative Adversarial Network

- Convergence et optimalité : minimisation sur G

$$\mathbb{E}_{x\sim P_X} \log\left[\frac{P_X(x)}{P_X(x)+P_G(x)}\right] + \mathbb{E}_{x\sim P_G} \log\left[\frac{P_G(x)}{P_X(x)+P_G(x)}\right]$$

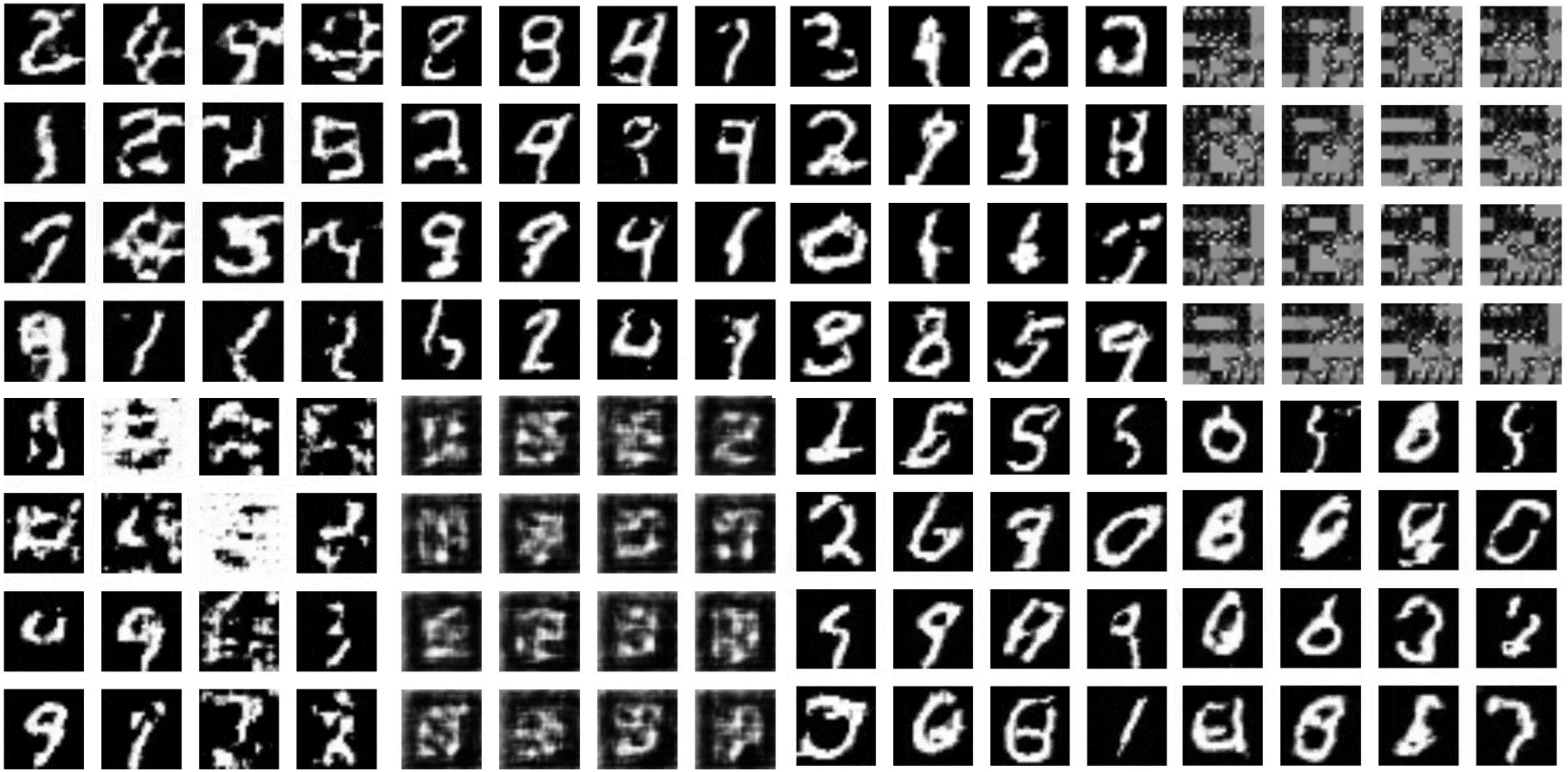
$$\mathbb{E}_{x\sim P_X} \log\left[\frac{P_X(x)^{\frac{1}{2}}}{(P_X(x)+P_G(x))^{\frac{1}{2}}}\right] + \mathbb{E}_{x\sim P_G} \log\left[\frac{P_G(x)^{\frac{1}{2}}}{(P_X(x)+P_G(x))^{\frac{1}{2}}}\right]$$

$$\mathbb{E}_{x\sim P_X} \log\left[\frac{P_X(x)}{\frac{(P_X(x)+P_G(x))}{2}}\right] + \mathbb{E}_{x\sim P_G} \log\left[\frac{P_G(x)}{\frac{(P_X(x)+P_G(x))}{2}}\right] - 2\log 2$$

$$KL\left(P_X \,||\, \frac{P_X+P_G}{2}\right) + KL\left(P_G \,||\, \frac{P_X+P_G}{2}\right) - \log 4 \quad = \quad JS\left(P_X \,||\, P_G\right) - \log 4$$

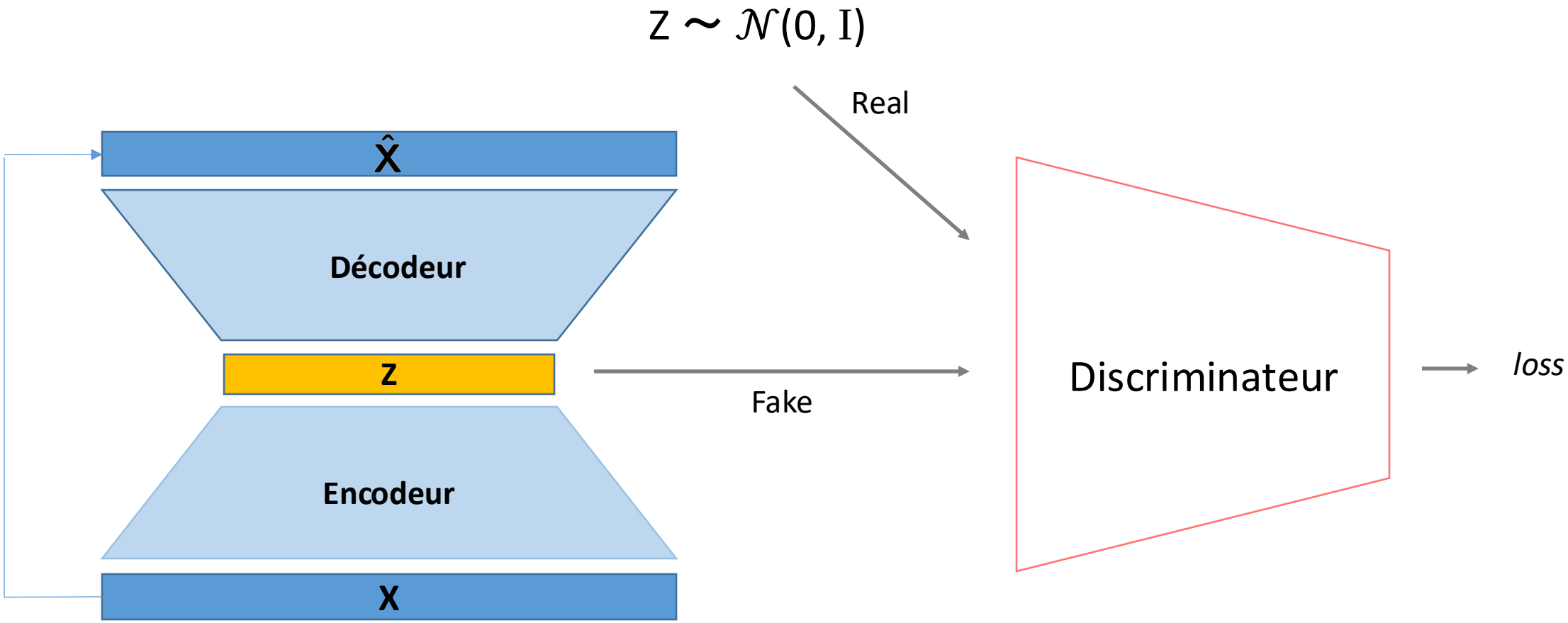
➔ Minimise la divergence de Jensen-Shannon entre P_X et P_G !

Entrainé sur MNIST (50000 images 32x32 pixels de chiffres manuscrits)



Adversarial Autoencoder

- [Makhzani, 2016]

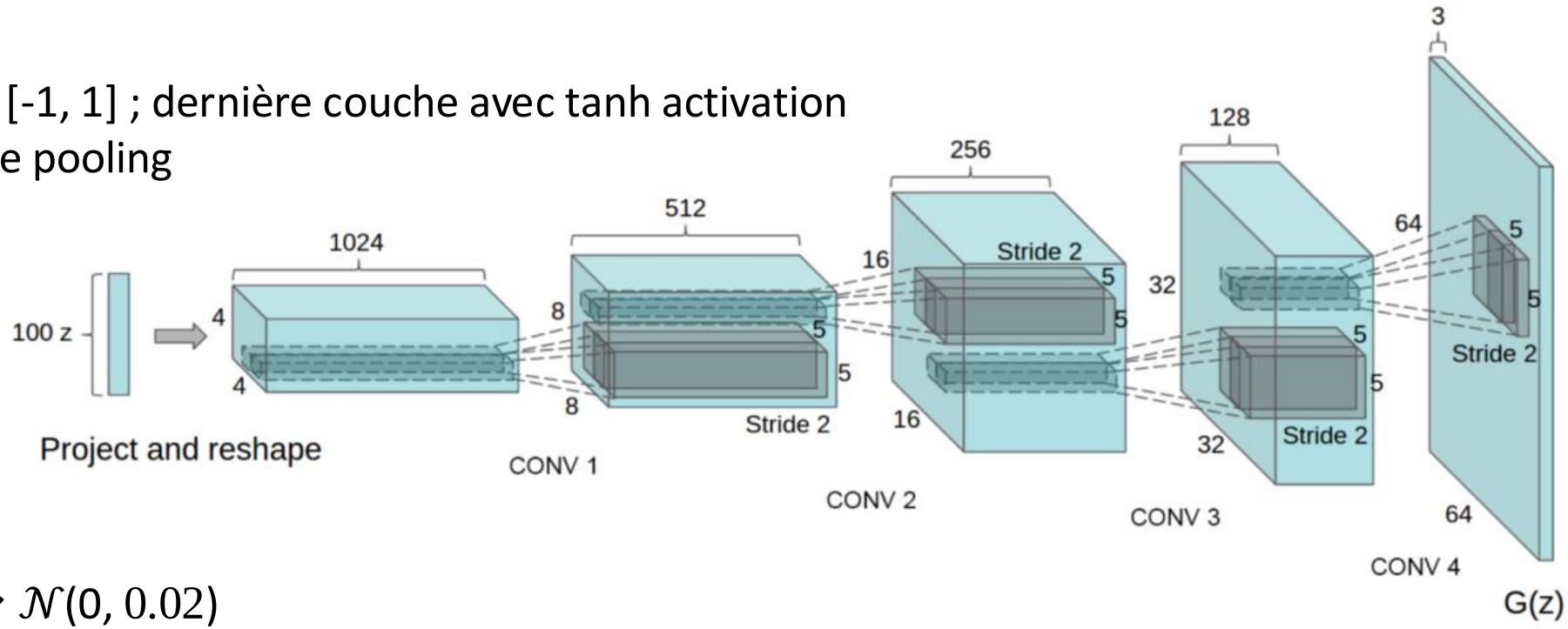


DCGAN

- [Radford, 2014]
- Architecture convolutionnelle dans le Générateur (et le Discriminateur)

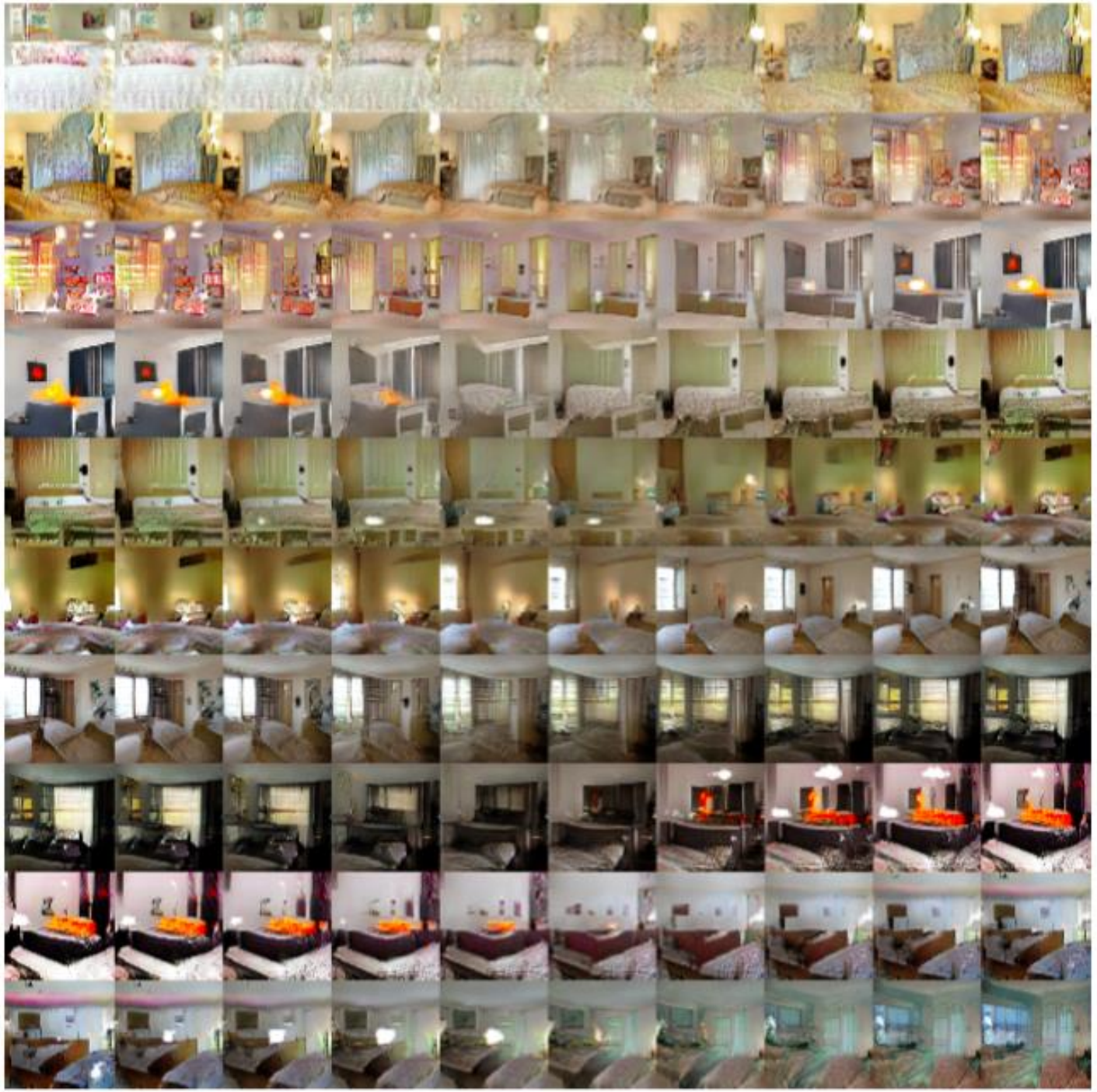
- images normalisées vers [-1, 1] ; dernière couche avec tanh activation
- stride convolution, pas de pooling
- batchNormalization
- leakyReLU(0.2)

- minibatch = 128
- Adam optimizer
  - momentum = 0.5
  - LR = 0.0002
- poids initialisés avec Z $\sim \mathcal{N}(0, 0.02)$



DCGAN

DCGAN



DCGAN

DCGAN






D’autres améliorations

- CGAN : Conditional GAN [Mirza, 2014]
  ⇒ Minimise la f-divergence de Jensen-Shannon entre P(X|Z,Y) et Q(X|Z,Y)

- LSGAN : Least-Squares GAN [Mao, 2016]
  ⇒ Minimise la f-divergence de Pearson-Chi2 entre P(X|Z) et Q(X|Z)

- WGAN : Wasserstein GAN [Arjovsky, 2017]
  ⇒ Optimise une distance de transport optimal entre P(X|Z) et Q(X|Z)

- ALI : Adversarially Learned Inference [Dumoulin, 2017]
  ⇒ Optimise la f-divergence de Jensen-Shannon entre P(X,Z) et Q(X,Z)
  ⇒ Q(X, Z) = Q(X) Q(X|Z) et P(X, Z) = P(Z) P(X|Z)
  ⇒ Apprend la distribution jointe (et la marginale)

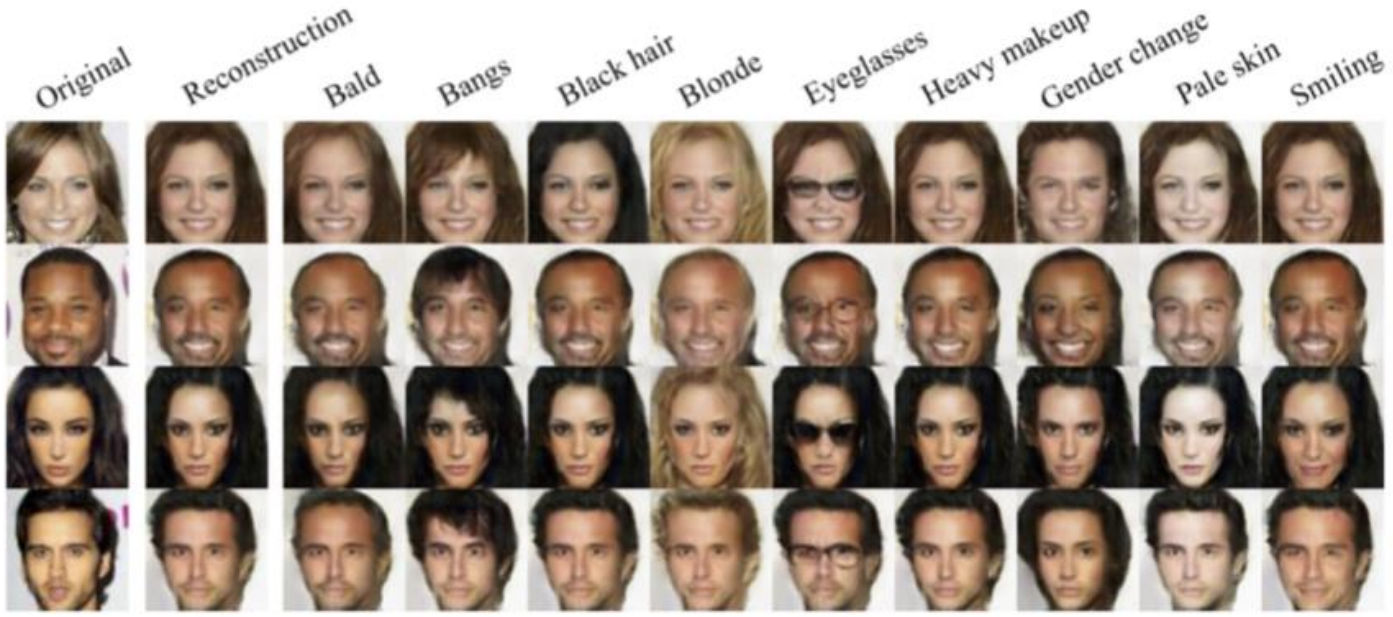
- Progressive Growing of GANs for Improved Quality, Stability, and Variation [Karras, 2018]
  - Génère des visages 1024x1024 pixels !

Conditional GAN

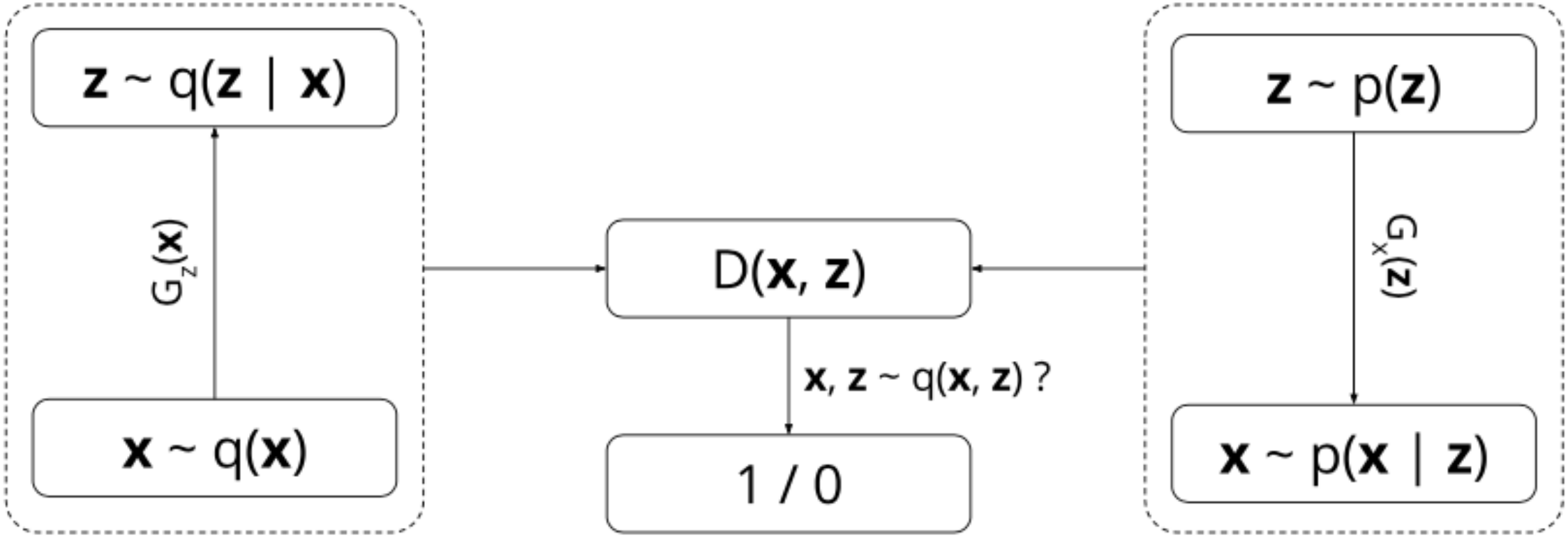
- [Mirza, 2014]

$$min_g max_d v(\theta_g, \theta_d) = \mathbf{E}_{x,y\sim p_{data}}\left[logD(x,y)\right] + \mathbf{E}_{z\sim p_z, y'\sim p_y}\left[log(1 - D(G(z,y'),y'))\right]$$

- ICGAN : Invertible Conditional GAN [Perarnau, 2016]

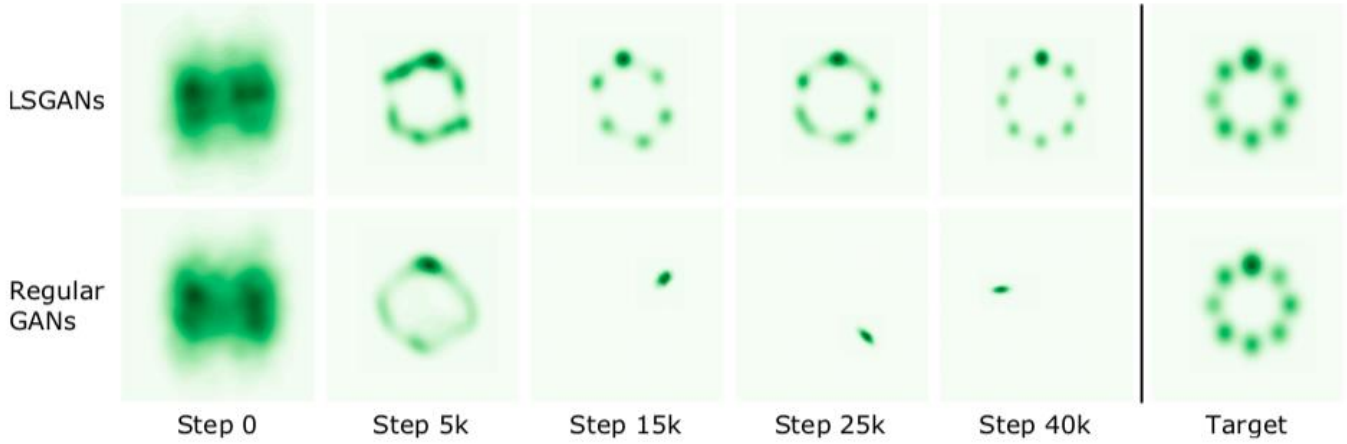


ALI [Dumoulin, 2017]



Apprentissage des GANs compliqué : « Mode Collapse »

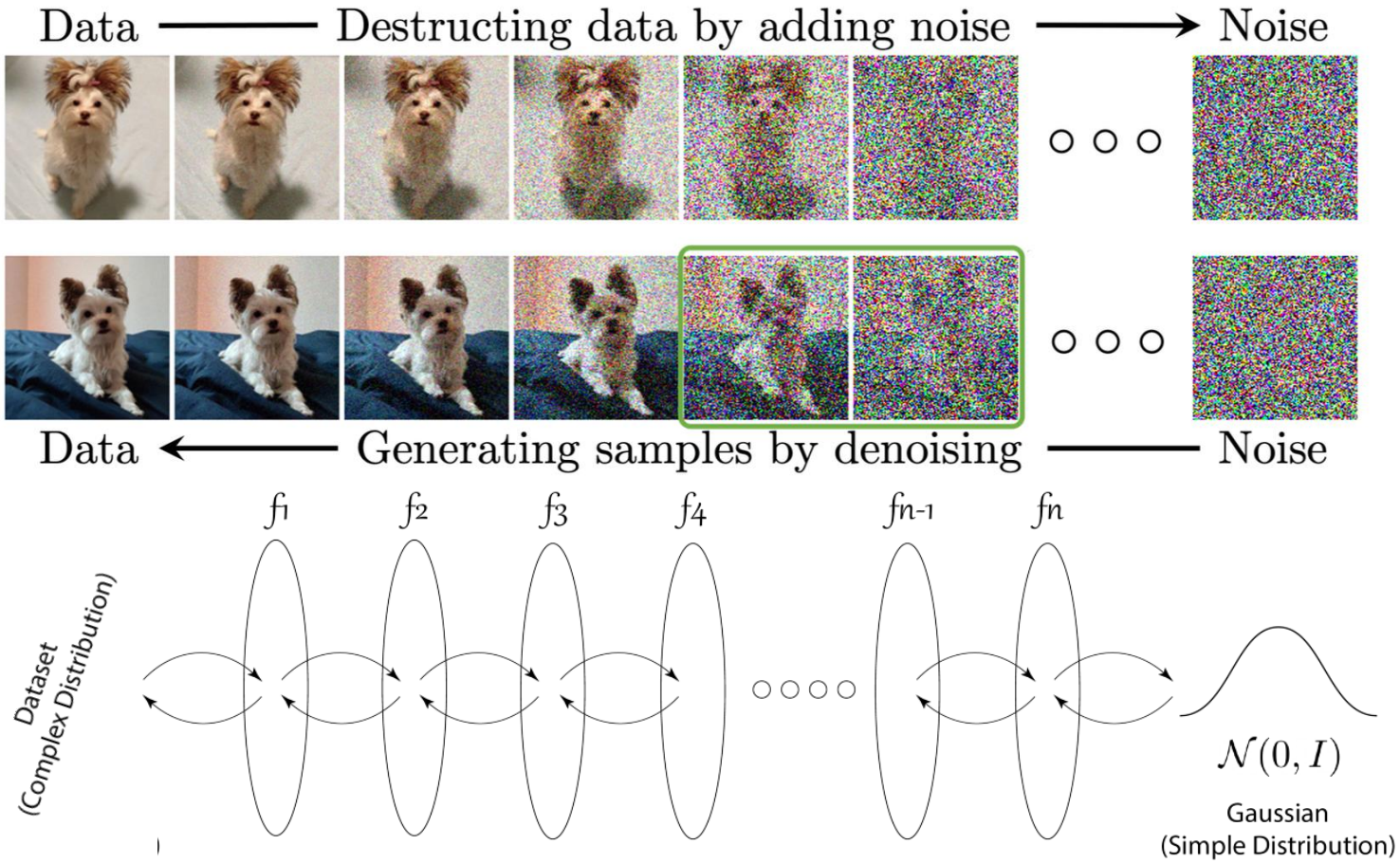
- Difficulté à apprendre une distribution à plusieurs modes
  - ➔ Le générateur produit des données peu variées
  - ➔ Le discriminateur a mal appris



- Comment le contourner ?
  - Experience Replay
  - Multiple GANs, ie: AdaGAN [Tolstikhin, 2017]
  - Minibatch Discrimination [Salimans, 2016]
  - Minibatch Standard Deviation [Karras, 2018]

Diffusion Models

→ Ajoute du bruit progressivement jusqu'à détruire la structure des données



Diffusion Models – 3 formulations, même chose

- Denoising Diffusion Probabilistic Models (DDPMs)
  - [Ho el al, NIPS'20] [Nichols et al, ICML'21] …
- Score-based Generative Models (SGMs)
  - [Song et al, NIPS'19-20]
- Stochastic Differential Equations (Score SDEs)
  - [Song et al, ICLR'20] [Song et al, NIPS'21]

- ICML 2015 : « Deep unsupervised learning using nonequilibrium thermodynamics », Sohl-Dickstein et al.

Diffusion Models - DDPMs [Nichols et al, ICML'21]

- Forward process :
  - $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$

où $\beta_t \in]0,1[$

  - $q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t) I)$

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$$

où $\epsilon \sim \mathcal{N}(0, I)$, $\alpha_t = 1 - \beta_t$ et $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$.

  - $q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \mu_t(\tilde{x_t}, x_0), \tilde{\beta}_t I)$

$$\mu_t(\tilde{x_t}, x_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha_{t-1}})x_t + \beta_t\sqrt{\bar{\alpha_{t-1}}}x_0}{1-\bar{\alpha}_t}$$

$$\tilde{\beta}_t = \frac{(1-\bar{\alpha_{t-1}})\beta_t}{1-\bar{\alpha}_t}$$

Diffusion Models - DDPMs [Nichols et al, ICML'21]

- Reverse process :

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

- Entrainement : $L_{simple} = E_{t,x_0,\epsilon}\left[||\epsilon - \epsilon_\theta(x_t, t)||^2\right]$

$$L_{alternative} = L_{simple} + 0.001 * L_{vlb}$$

$$L_{vlb} = \sum_{t=0}^{T} L_t$$
$$L_T = D_{KL}(q(x_T|x_0) \,||\, p(x_T))$$
$$L_{t-1} = D_{KL}(q(x_{t-1}|x_t, x_0) \,||\, p_\theta(x_{t-1}|x_t))$$
$$L_0 = -\log p_\theta(x_0|x_1)$$

Latent Diffusion Model (Stable Diffusion)

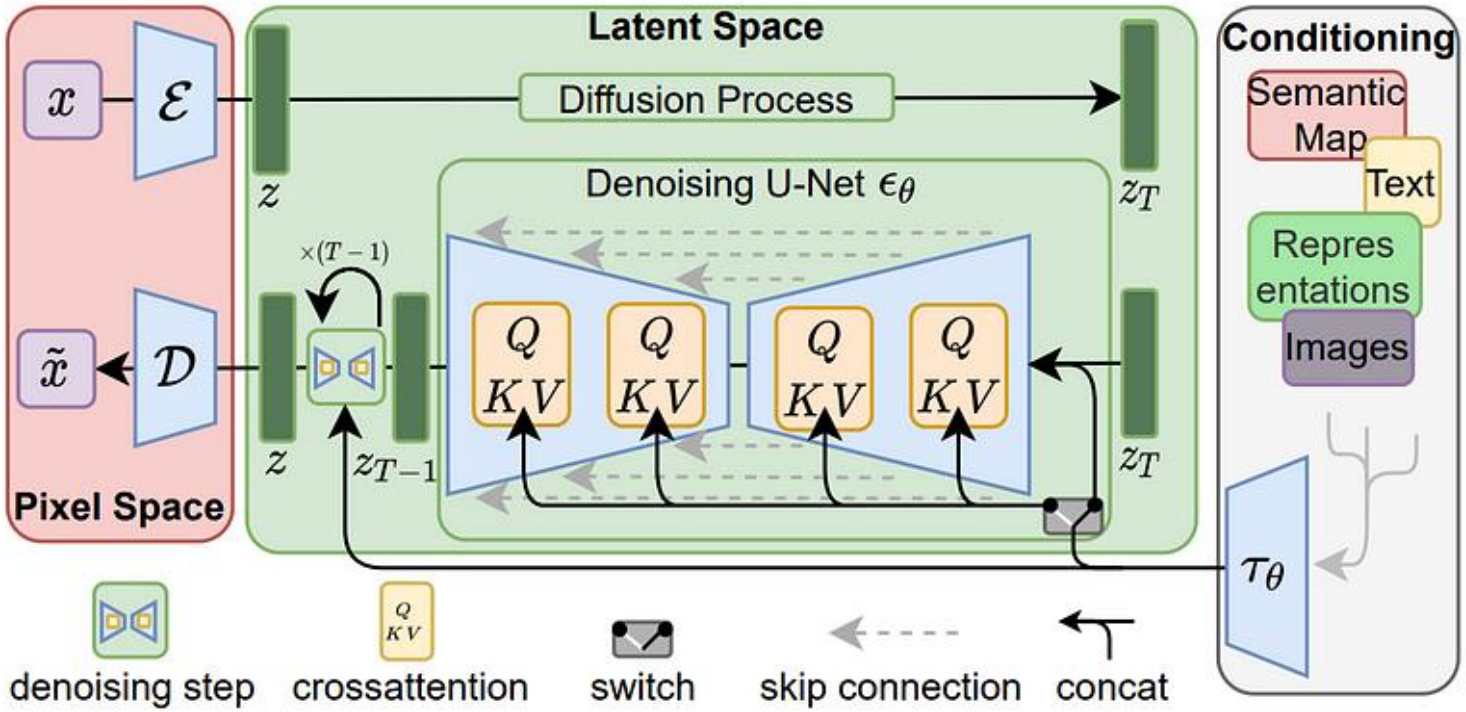


Figure 3. We condition LDMs either via concatenation or by a more general cross-attention mechanism.

- CVPR 2022 : « High-Resolution Image Synthesis with Latent Diffusion Models », Rombach et al.

Un mot sur l'évaluation

- Evaluation qualitative
  - « à l'œil » !
  - À plus ou moins grande échelle..

- Evaluation quantitative
  - MS-SSIM, PSNR
  - Inception Score : $\mathbf{IS}(G) = \exp\left(\mathbb{E}_{\mathbf{x} \sim p_g} D_{KL}(p(y|\mathbf{x}) \parallel p(y)) \right)$
  - FID
  - …